Name: **Sourojit Saha**
Andrew id: **sourojis**

Course: **16-782 Planning and Decision Making in Robotics**
Assignment: **Assignment 3**

1. Algorithm

This implementation for path planning creates implicit graphs as the search progresses. Each node is a class, which stores the following information:

- Unique identifier for itself (self_id), a hash
- Unique identifier of its parent node (parent_id), a hash
- The f value
- The g value
- The h value
- An unordered set that stores the state of the node
- The action which when applied to its parent node, creates this node

The data structures used were:

- A priority queue for the openlist. It stored the node. The node with the least g value was at the top.
- An unordered map for closedlist. It stored the hash for the node and the node itself.
- An unordered map for hash-table. It stored all the hashes of all the nodes that were created and the nodes themselves. The hash-table was used for backtracking.

The flow of the algorithm is as follows:

- The start node was created and inserted into the openlist. Then the element with the smallest f value was extracted from the openlist. Its state was checked with the goal state and inserted into the closedlist. If this node state was found to satisfy all the condition for goal node, this was taken as "goal found", and the A star algorithms ended. If this node did not satisfy the goal conditions, then the valid actions that can be done on this state were computed. Using this node and the valid actions, the child nodes were created. The heuristics of the child nodes were calculated. The f and g values were updated and inserted in openlist, if not already in openlist, else the f,g,h values were updated.
- For each actions that was performed, the g value was increased by 1.
- The heuristic used was the number of goal conditions that were not satisfied by that node.
- Once the final goal state was available, the path was back-tracked to get the full path from the start state to the final goal state. The hash-table was used for backtracking

2. Results

For the applied heuristic, as mentioned in previous section, the plans generated for the tree test cases are as follows:

For Blocks.txt:
Plan:
MoveToTable(A,B)
Move(C,Table,A)
Move(B,Table,C)

For BlocksTriangle.txt:
Plan:
MoveToTable(T0,B0)
MoveToTable(T1,B3)
Move(B0,B1,B3)
MoveToTable(B0,B3)
Move(B1,B4,B3)
Move(B0,Table,B1)
Move(T1,Table,B0)

For FireExtinguisher.txt:
Plan:
MoveToLoc(A,F)
MoveToLoc(F,E)
MoveToLoc(E,C)
MoveToLoc(C,W)
MoveToLoc(W,D)
MoveToLoc(D,B)
LandOnRob(B)
MoveTogether(B,F)
MoveTogether(F,W)
FillWater(Q)
MoveTogether(W,A)
MoveTogether(A,C)
MoveTogether(C,D)
MoveTogether(D,B)
MoveTogether(B,E)
MoveTogether(E,F)
TakeOffFromRob(F)
PourOnce(F)
LandOnRob(F)
MoveTogether(F,B)
MoveTogether(B,E)
MoveTogether(E,W)
FillWater(Q)
Charge(Q)
MoveTogether(W,C)

MoveTogether(C,A)
MoveTogether(A,F)
TakeOffFromRob(F)
PourTwice(F)
LandOnRob(F)
MoveTogether(F,B)
MoveTogether(B,A)
MoveTogether(A,D)
MoveTogether(D,E)
MoveTogether(E,W)
FillWater(Q)
MoveTogether(W,C)
MoveTogether(C,A)
MoveTogether(A,D)
MoveTogether(D,E)
MoveTogether(E,B)
MoveTogether(B,F)
Charge(Q)
TakeOffFromRob(F)
PourThrice(F)

The summary of the results from the three test cases as given in the assignment are as follows:

Environment	Time Taken	No. of steps	States Explored
Blocks.txt	0.006114s	3	17
BlocksTriangle.txt	0.583338s	7	389
FireExtinguisher.txt	4.6421s	45	818

For the case of no heuristic, the computer ran out of memory and never found a solution.

3. Instructions to compile

Here are the steps to compile:

- Compile the planner.cpp file `g++ planner3.cpp -o planner.out`
- Run the compiled file `./planner.out <name of description file>`
- Example `./planner.out Blockes.txt`